Applied Machine Learning

Mini-Project Report

-KAUSHIK PARVATHANENI(kaparvat@iu.edu)

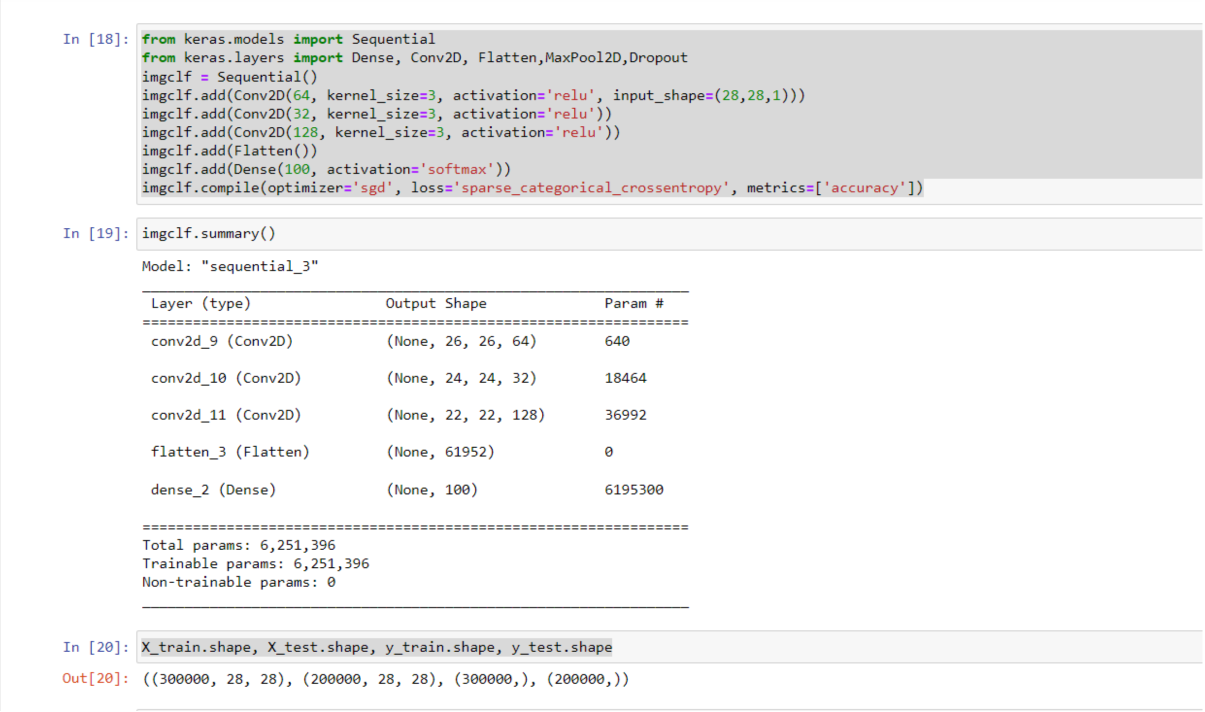
1)Project Description and Formulation of problem:

-A dataset of images is given which is of 500000 images and a testing dataset of 100000 images as a pickle file. We need to create a model such that the model detects images to their respective labels.

-I have used Convolutional neural network to classify the images here.

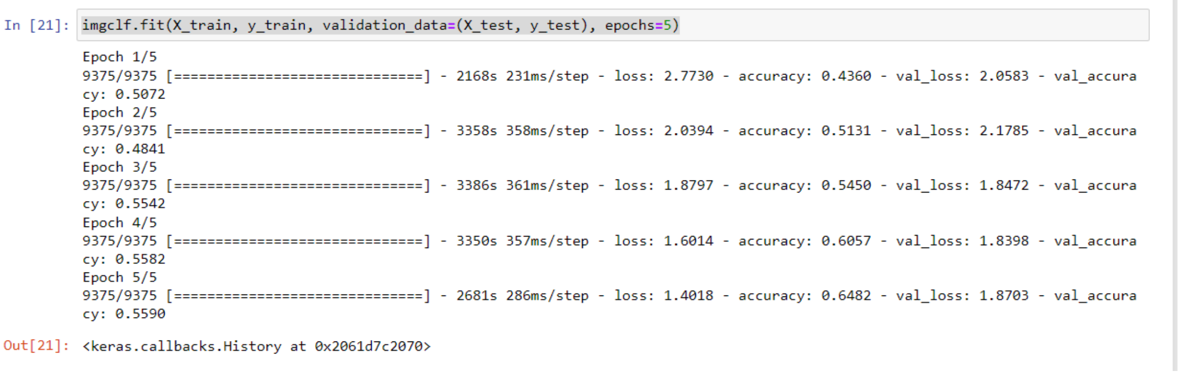
2)Preprocessing, Assumption and Model Design:

-First I have read the pickle files and imported the data. Then by using train_test_split we splitted the data and target values into the test and train values. Then I built the CNN model. I have performed the model as followed and I have also showed the model summary.

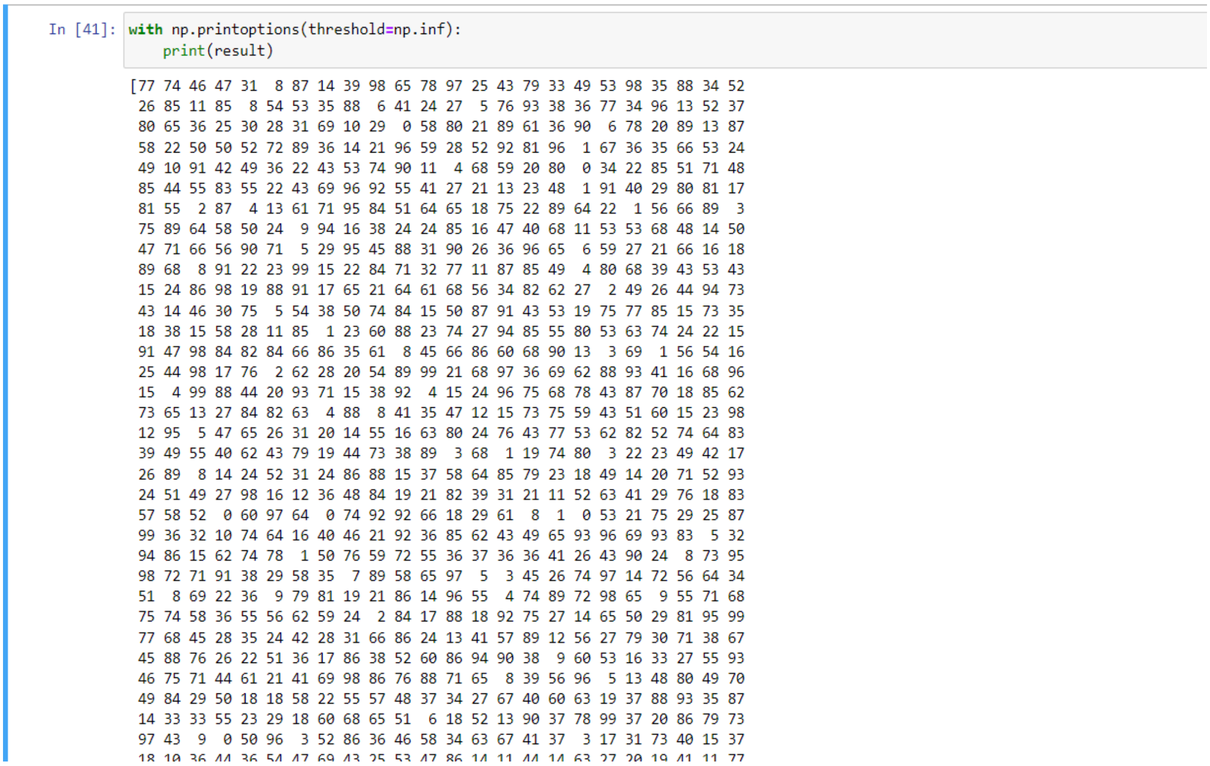


```
In [18]: from keras.models import Sequential
         from keras.layers import Dense, Conv2D, Flatten,MaxPool2D,Dropout
         imgclf = Sequential()
         imgclf.add(Conv2D(64, kernel_size=3, activation='relu', input_shape=(28,28,1)))
         imgclf.add(Conv2D(32, kernel_size=3, activation='relu'))
         imgclf.add(Conv2D(128, kernel_size=3, activation='relu'))
         imgclf.add(Flatten())
         imgclf.add(Dense(100, activation='softmax'))
         imgclf.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

In [19]: imgclf.summary()

         Model: "sequential_3"
         _________________________________________________________________
          Layer (type)                Output Shape              Param #
         =================================================================
          conv2d_9 (Conv2D)           (None, 26, 26, 64)        640

          conv2d_10 (Conv2D)          (None, 24, 24, 32)        18464

          conv2d_11 (Conv2D)          (None, 22, 22, 128)       36992

          flatten_3 (Flatten)         (None, 61952)             0

          dense_2 (Dense)             (None, 100)               6195300

         =================================================================
         Total params: 6,251,396
         Trainable params: 6,251,396
         Non-trainable params: 0
         _________________________________________________________________

In [20]: X_train.shape, X_test.shape, y_train.shape, y_test.shape

Out[20]: ((300000, 28, 28), (200000, 28, 28), (300000,), (200000,))
```

3)Result:



```
In [21]: imgclf.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=5)

         Epoch 1/5
         9375/9375 [==============================] - 2168s 231ms/step - loss: 2.7730 - accuracy: 0.4360 - val_loss: 2.0583 - val_accura
         cy: 0.5072
         Epoch 2/5
         9375/9375 [==============================] - 3358s 358ms/step - loss: 2.0394 - accuracy: 0.5131 - val_loss: 2.1785 - val_accura
         cy: 0.4841
         Epoch 3/5
         9375/9375 [==============================] - 3386s 361ms/step - loss: 1.8797 - accuracy: 0.5450 - val_loss: 1.8472 - val_accura
         cy: 0.5542
         Epoch 4/5
         9375/9375 [==============================] - 3350s 357ms/step - loss: 1.6014 - accuracy: 0.6057 - val_loss: 1.8398 - val_accura
         cy: 0.5582
         Epoch 5/5
         9375/9375 [==============================] - 2681s 286ms/step - loss: 1.4018 - accuracy: 0.6482 - val_loss: 1.8703 - val_accura
         cy: 0.5590

Out[21]: <keras.callbacks.History at 0x2061d7c2070>
```



```
In [41]: with np.printoptions(threshold=np.inf):
             print(result)

         [77 74 46 47 31  8 87 14 39 98 65 78 97 25 43 79 33 49 53 98 35 88 34 52
          26 85 11 85  8 54 53 35 88  6 41 24 27  5 76 93 38 36 77 34 96 13 52 37
          80 65 36 25 30 28 31 69 10 29  0 58 80 21 89 61 36 90  6 78 20 89 13 87
          58 22 50 50 52 72 89 36 14 21 96 59 28 52 92 81 96  1 67 36 35 66 53 24
          49 10 91 42 49 36 22 43 53 74 90 11  4 68 59 20 80  0 34 22 85 51 71 48
          85 44 55 83 55 22 43 69 96 92 55 41 27 21 13 23 48  1 91 40 29 80 81 17
          81 55  2 87  4 13 61 71 95 84 51 64 65 18 75 22 89 64 22  1 56 66 89  3
          75 89 64 58 50 24  9 94 16 38 24 24 85 16 47 40 68 11 53 53 68 48 14 50
          47 71 66 56 90 71  5 29 95 45 88 31 90 26 36 96 65  6 59 27 21 66 16 18
          89 68  8 91 22 23 99 15 22 84 71 32 77 11 87 85 49  4 80 68 39 43 53 43
          15 24 86 98 19 88 91 17 65 21 64 61 68 56 34 82 62 27  2 49 26 44 94 73
          43 14 46 30 75  5 54 38 50 74 84 15 50 87 91 43 53 19 75 77 85 15 73 35
          18 38 15 58 28 11 85  1 23 60 88 23 74 27 94 85 55 80 53 63 74 24 22 15
          91 47 98 84 82 84 66 86 35 61  8 45 66 86 60 68 90 13  3 69  1 56 54 16
          25 44 98 17 76  2 62 28 20 54 89 99 21 68 97 36 69 62 88 93 41 16 68 96
          15  4 99 88 44 20 93 71 15 38 92  4 15 24 96 75 68 78 43 87 70 18 85 62
          73 65 13 27 84 82 63  4 88  8 41 35 47 12 15 73 75 59 43 51 60 15 23 98
          12 95  5 47 65 26 31 20 14 55 16 63 80 24 76 43 77 53 62 82 52 74 64 83
          39 49 55 40 62 43 79 19 44 73 38 89  3 68  1 19 74 80  3 22 23 49 42 17
          26 89  8 14 24 52 31 24 86 88 15 37 58 64 85 79 23 18 49 14 20 71 52 93
          24 51 49 27 98 16 12 36 48 84 19 21 82 39 31 21 11 52 63 41 29 76 18 83
          57 58 52  0 60 97 64  0 74 92 92 66 18 29 61  8  1  0 53 21 75 29 25 87
          99 36 32 10 74 64 16 40 46 21 92 36 85 62 43 49 65 93 96 69 93 83  5 32
          94 86 15 62 74 78  1 50 76 59 72 55 36 37 36 36 41 26 43 90 24  8 73 95
          98 72 71 91 38 29 58 35  7 89 58 65 97  5  3 45 26 74 97 14 72 56 64 34
          51  8 69 22 36  9 79 81 19 21 86 14 96 55  4 74 89 72 98 65  9 55 71 68
          75 74 58 36 55 56 62 59 24  2 84 17 88 18 92 75 27 14 65 50 29 81 95 99
          77 68 45 28 35 24 42 28 31 66 86 24 13 41 57 89 12 56 27 79 30 71 38 67
          45 88 76 26 22 51 36 17 86 38 52 60 86 94 90 38  9 60 53 16 33 27 55 93
          46 75 71 44 61 21 41 69 98 86 76 88 71 65  8 39 56 96  5 13 48 80 49 70
          49 84 29 50 18 18 58 22 55 57 48 37 34 27 67 40 60 63 19 37 88 93 35 87
          14 33 33 55 23 29 18 60 68 65 51  6 18 52 13 90 37 78 99 37 20 86 79 73
          97 43  9  0 50 96  3 52 86 36 46 58 34 63 67 41 37  3 17 31 73 40 15 37
          18 10 36 44 36 54 47 69 43 25 53 47 86 14 11 44 14 63 27 20 19 41 11 77
```

-The complete final result is stored in “project_kaparvat.txt”.

4)How can we improve our accuracy:

- By trying different optimizers

- We can try increasing the layers in the model by keeping different size of filters which may be suitable.

-Better pre-processing of the data.